CSCI 341–Fall 2024: Lecture Notes
Set 4: NFAs

Edward Talmage

September 27, 2024

Motivation: We want to use the ideas from the proof that regular languages are closed under union to show that they are also closed under concatenation.

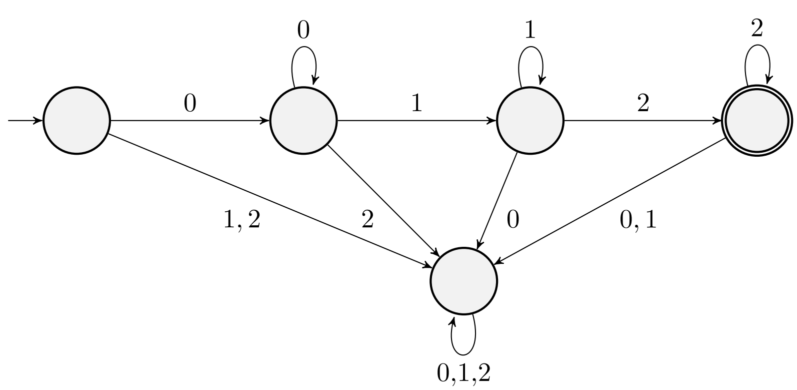
- We want to run through M_1, then go to the start state of M_2, and run through it. If we get to an M_2 accept state, then we should accept.
- Issues:
  - When do we stop in M_1 and start in M_2?
  - How do we get from M_1 to M_2 without consuming a symbol?

1 Non-deterministic Finite Automata

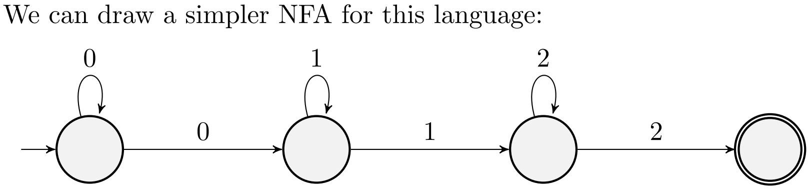
We will introduce a new type of automaton, a *Non-Deterministic Finite Automaton* or *NFA*. An NFA

- is still an automaton: process an input string by moving between states;
- is still finite, memory is still just the state set;
- has transitions which are not deterministic.
  - Determinism is when behavior is completely determined: same input yields same output.
  - Now, same input may yield multiple (or no) possible outputs.
- Example:

Exercise: Design a DFA that recognizes $L = \{0^i 1^j 2^k \mid i, j, k \geq 1\}$.



We can draw a simpler NFA for this language:

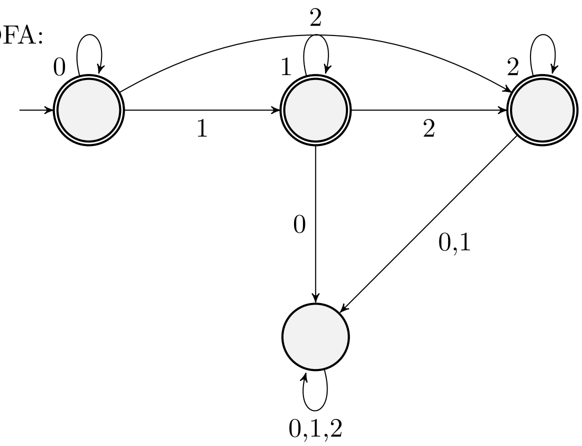


- Examples:

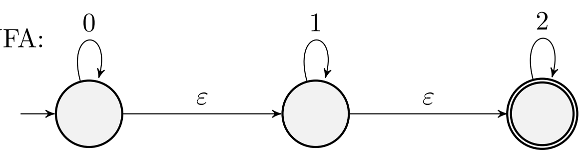
> **Exercise:** Draw a DFA and an NFA for the language $L_2 = \{0^i 1^j 2^k \mid i, j, k \geq 0\}$

.

DFA:



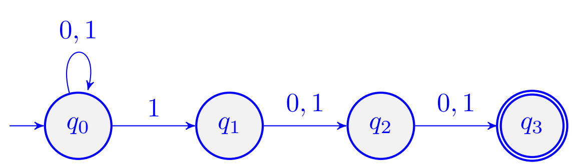
NFA:



New rules:

- Each step will spawn new processes, which each traverse the machine, one for each edge for the next symbol from the current state.
- Accept if **any** process ends in an accept state. Reject if **all** processes end in non-accept states.
- We can have edges which do not consume a symbol from the input. Instead, they treat the empty string as the next character.
  - This fits our intuitive definition of non-determinism, as there's now more than one thing we could do on no input: stay put or follow an empty-string edge.
- If there's no valid edge to follow, a process dies, never accepting.

> **Exercise:** What language does the following NFA recognize?



2 Formal Definitions

Definition 1. A *Non-deterministic Finite Automaton*, or *NFA*, is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set of states
- Σ is the alphabet
- $\delta : Q \times (\Sigma \cup \{\varepsilon\} \to \mathcal{P}(Q)$ is the transition function
- $q_0 \in Q$ is the start state

- $F \subseteq Q$ is the set of accept states

Notes:

- Ongoing, we will use Σ_ε to denote $\Sigma \cup \{\varepsilon\}$.
- Note that $\delta(q, a)$ may be $\emptyset$. We don't draw these transitions. This is basically a `kill -9` for a process.

Exercise: Represent our NFA for $\{0^i 1^j 2^k \mid i, j, k \geq 0\}$ by set notation.

Definition 2. An NFA N *accepts* a string w iff there exist $a_1, a_2, \ldots, a_n \in \Sigma_\varepsilon$ and $r_1, \ldots, r_{n+1} \in Q$ s.t.

1. $w = a_1 a_2 \cdots a_n$ (Note: This is non-trivial, since some a_i's may be ε!)
2. $r_1 = q_0$
3. $r_{n+1} \in F$
4. $\forall 1 \leq i \leq n, \delta(r_i, a_i) \ni r_{i+1}$

We also need to consider how to apply the transition function to sets of states S, instead of just single states, since there may be many processes in different states at the same time.

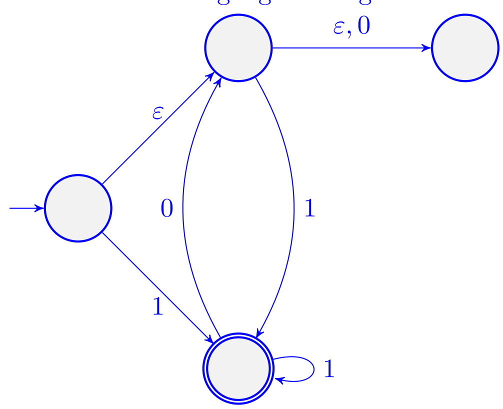
- $E(q) = \{r \in Q \mid \exists \text{ a path} q \rightsquigarrow r \text{ using only } \varepsilon\text{-transitions}\}$
- $E(S) = \bigcup_{q \in S} E(q)$ (These are the *epsilon closures* of a state or set of states.)
- $\delta(S, a) = \bigcup_{q \in S} \delta(q, a)$

Definition 3. Extend δ to accept strings: $\hat{\delta} : \mathcal{P}(Q) \times \Sigma^* \to \mathcal{P}(Q)$

1. $\hat{\delta}(S, \varepsilon) = E(S)$
2. $\forall u \in \Sigma^*, a \in \Sigma : \hat{\delta}(S, ua) = E(\delta(\hat{\delta}(S, u), a))$
    - Alt.: $\hat{\delta}(S, au) = \hat{\delta}(\delta(S, a), u) = \hat{\delta}(E(\delta(S, a)), u)$

N accepts w iff $\hat{\delta}(\{q_0\}, w) \cap F \neq \emptyset$.

Exercise: For the following NFA, give the ε-closure of each state. Try running it on a few strings and determine what language t recognizes.



$L = \{w \mid w \text{ neither starts with 0 nor contains 00}\}$.